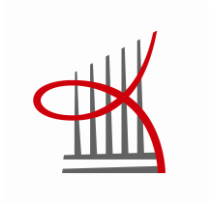


RELAATIOTIETOKANNAN DOKUMENTOINTI

Petri Ahola

Opinnäytetyö
Kesäkuu 2010
Tietojenkäsittelyn koulutusohjelma
Terveys- ja sosiaalialan tietohallinta
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Terveysalan tietohallinta

AHOLA, PETRI:
Relaatiotietokannan dokumentointi

Opinnäytetyö 27 sivua
Kesäkuu 2010

Sähköiset tietokannat ovat olennainen osa arkea. Viimeiset kaksi vuosikymmentä ovat vähentäneet paperitöitä tietokantojen kehittyessä. Erilaiset tietokannat toimittavat ja pitävät yllä tietoja aina matkapuhelimen toiminnoista valtion henkilötietokantaan. Tietokantaohjelmistot ja -ohjelmointikielet ovat kehittyneet monipuolistaen tietokantojen käyttömahdollisuuksia.

Isompien tietokantojen toimintaa päivittämään ja ylläpitämään tarvitaan asiantuntijoita sekä tietokantaohjelmoinnin että sovellussuunnittelun alalta. Usein tietokannan toiminta ja itse tietojen ylläpito ovat täysin eri vastuualueita. Tietokantojen toiminnan ylläpitäminen edellyttää tietojen tarkastelua. Kehitys- ja ylläpitotyön ratkaisut jäävät usein salaisiksi tietojen ollessa usein luottamuksellista. Tämän takia eri tietokannoissa käytettyjä ratkaisuja ei päästä hyväksikäyttämään muilla aloilla ennen kuin ratkaisut löytävät tiensä itse tietokantaohjelmien päivityksiin.

Tämän työn tarkoituksena oli perehtyä tietokantojen ongelmien dokumentointiin ja dokumentoinnin merkitykseen tietokannan toiminnan ylläpitoon ja kehitykseen. Teoriaosassa käsitellään enimmäkseen yleisimpiä ongelmakohtia ja ongelmia, jotka ovat selvästi huomattavissa. Jokaisesta tietokannasta on löydettävissä kyseisentyyppiselle tietokannalle ominaisia ja jopa uniikkeja virheitä. Tämän takia ei lähdetty hakemaan yksityiskohtaista tarkastelua, vaan keskityttiin loogisesti yleisimpiin virheisiin. Yleisen tarkastelun perusteella saadaan tarvittavaa tietoa, jonka avulla voidaan päätellä tietokannan yksilöllisempiäkin ongelmakohtia.

Dokumentoinnin avulla tietokannan rakenne saadaan selville. Ongelmien dokumentointi tuo esille rakenteelliset puutteet ja mahdolliset heikot kohdat tietojen oikeellisuuden osalta. Dokumentoinnin avulla tietokannan rakenteesta saadaan helpommin ylläpidettävä ja päivittämisestä saadaan luotettavampaa. Ylläpitämisen ja päivittämisen helpottuessa työmäärä vähenee. Välillisesti hyvin hoidettu dokumentointi sekä vähentää kustannuksia, että tekee rakenteesta vakaamman, monipuolisemman ja joustavamman.

Opinnäytetyö selkeytti dokumentoinnin merkitystä ja tarvetta varsinkin keskikokoisten ja suurten tietokantarakenteiden osalta. Se antoi myös hyvän pohjan ymmärtää tietokantarakenteissa huomioitavia ongelmakohtia ja dokumentoinnin tarvetta.

Asiasanat: relaatio, tietokanta, dokumentointi

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in Business Information Technology
Option of Data Management in the Field of Health Care

AHOLA, PETRI:
Dokumentation of Relational Database

Thesis 27 pages
June 2010

Digital databases are an essential part of every day life. During the last two decades the amount of paperwork has been decreased due to the development of databases. Different kinds of databases deliver and sustain data from mobile phone functions up to government's personnel database. Database softwares and programming languages have developed and diversified the database affordances.

Extensive databases require experts in the field of database programming and application design to patch and maintain the database functions. Very often database functions and data maintenance belong to completely different areas of responsibility. Analysis of the data is required when upholding the database functions. Due to the confidential nature of data, development and maintenance solutions are often hidden. Because of that, solutions used in different databases can not be utilized in other domains until the solutions find their way to the database software updates.

The purpose of this thesis is to get acquainted with documenting the problems in databases and the consequences of documentation in maintaining and developing databases. The most common problems that are distinctly detected are discussed in the theory section.

In every database there are exceptions that are characteristic and even unique to the database in question. Because of that, this thesis is not concentrating on detailed analysis but logically the most common exceptions. The general analysis gives information that can be used to conclude more distinctive problems in database.

The structure of a database is resolved by documentation. Documenting problems highlights the structural deficiencies and possible weak spots as for the validity of data. Documentation helps creating a database structure that is easier to maintain and more reliable to update. The workload decreases as maintaining and updating becomes easier. A vicariously done documentation reduces costs and makes the database structure more stable, diverse and flexible.

This thesis clarifies the meaning and need of documentation, especially in case of medium-sized and extensive database structures. It also gives a good basis to understand database structure related problems that need to be taken into consideration and also the need of documentation.

Keywords: relation, database, documentation

SISÄLLYS

1 JOHDANTO

Dokumentoinnin perustavoite on antaa tietokannan omistajalle käsitys tietokannan senhetkisestä tilasta. Tietokanta dokumentoidaan aina tiettyyn ajankohtaan pyäsytetystä tietokannasta. Pysäytetystä tietokannasta voidaan puhua tietokannan kuvana. Kuvan dokumentointi ei ole täysin ajankohtainen tietojen kerääntymisen näkökulmasta, vaan se keskittyy tarkoituksenmukaisesti rakenteellisiin, pitkällä aikavälillä tapahtuviin, ilmiöihin.

Dokumentin antaman informaation pohjalta voidaan havaita tietokannassa esiintyviä virheitä, puutteita ja kehitystarpeita. Tarkoitus on tutkia dokumentoinnin antaman informaation pohjalta, ovatko ongelmat peräisin tietokannan rakenteellisista ratkaisuista, käyttäjävirheistä vai tietorajoitteiden puutteellisuudesta.

Virheisiin voidaan lukea kaikki selkeistä käytössä huomattavista viite-eheys-ongelmista aina tietokantaan tallennettuihin kirjoitusvirheisiin tai asiattomuuksiin. Puutteet ja kehitystarpeet ovat usein toisistaan riippuvaisia. Kehitystarpeet ja -pyynnöt ovat yleisesti käyttäjälähtöisiä, käyttäjälle näkyviä, tietokannan puutteita. Puutteita huomattaessa päädytään lähes aina löytämään myös muita rakenteellisia virheitä.

Opinnäytetyössä keskitytäänkin enemmän rakenteellisten ongelmien löytämiseen tietokannasta ja niiden dokumentointiin. Tarkoituksena on kartoittaa käyttäjälle näkymättömiä puutteita, jotka mahdollistavat tietokannan kuormittumisen pitkällä aikavälillä ja näin ollen hidastavat tietokannan toimivuutta. Virheiden kartoitus puolestaan auttaa tietokannan rakenteen korjaamista, päivittämistä ja ylläpitoa. Yleinen dokumentaatio auttaa uusien ominaisuuksien lisäämistä ja esittelemistä olemassa olevaan tietokantakokonaisuuteen. Esimerkkityökaluina käytetään Oracle SQL Developer -tietokantatyökalua ja Oracle Database 10g -tietokantapalvelinta. Esimerkkejä ei voitu salassapitovaatimusten vuoksi tehdä tuotantoympäristössä olevasta tietokannasta, joten opinnäytetyössä aihetta lähestytään itsenäisenä tarkasteluna toimeksiantajan ehdottamasta aiheesta. Toimeksiantaja ei halunnut nimeään julki opinnäytetyössä.

2 RELAATIOTIETOKANTA

Tietovarastolle käytetään tietotekniikassa termiä tietokanta. Tietokanta sisältää tietoja, joilla on jokin yhteinen tekijä tai yhteys toisiinsa. Yksinkertaisimpina esimerkkeinä voidaan listata puhelinnumeromuistio ja kalenteri. Yleisesti tietokannat ovat hyvin rajattuja omiin kohteisiinsa. Puhelinnumeromuistiossa on vain käyttäjälle tarpeellisia numeroita ja kalenterissa ne päivämäärät, jotka ovat käyttäjälle tärkeitä.

Relaatiotietokannan ero vanhaan kortistomalliseen, manuaalisestikin käytettävään, tietokantaan on nimensä mukaisesti relaatiot. Relaatio, suhde tai yhteys, tarkoittaa tietokannassa tietojen välistä yhteyttä toisiinsa. Relaatiotietokanta voidaan ymmärtää useana kortistona, joista voidaan noukkia kaikki tarvittava tieto samasta paikasta. Tietoja voidaan myös talllentaa yhdestä paikasta moneen eri kortistoon. Tietokannoissa kortistoa kutsutaan tauluksi ja korttia tietueeksi. Tietueen osat on jaettu taulussa sarakkeisiin eli kenttiin.

Relaatiomallisen tietokannan hyötyjä ovat relaatioiden sallima monipuolisuus ja joustavuus tiedon käsittelyssä. Yhden taulun sisältämää tietoa voidaan käyttää moneen eri tarkoitukseen monessa eri paikassa. Tiedot löytyvät kuitenkin aina samasta paikasta. Puhtaasti tiedon käsittelyn kannalta relaatiomallinen tietokanta on helpoin tapa ylläpitää ja käyttää massiivisia tietokokonaisuuksia.

Relaatiomallista saatavat hyödyt mahdollistavat ominaisuudet voidaan myös nähdä relaatiomallin haittoina. Tietojen jakaminen moneen eri paikkaan johtaa tietojen käytön lisääntymiseen. Tietojen useat relaatiot lisäävät tietojen korruptoitumisriskiä käyttäjien tekemien virheiden tai rakenteellisten virheiden johdosta. Relaatiotietokannan useiden tietoliikenneyhteyksien vuoksi dokumentointi ja ongelmien kartoittaminen on haastavaa mutta sitäkin tärkeämpää.

2.1 Tietokantojen historiaa

Tietokantojen juuret voidaan johtaa aina 1950-luvulle, jolloin tietokannat koostuivat yhdessä paikassa sijaitsevista yksittäisistä tiedostoista. Tietokannat mahdollistivat tuolloin tietdostojen järjestämisen ja ylläpidon. Myöhemmin samalla vuosikymmenellä kehitettiin General Electronic Companyn Mark 1 -raporttigeneraattori ja lajittelurutiini IBM:n 702 -koneeseen. Näiden avulla saatiin tiedostoista poimittua halutut tiedot raporteille. Tietojen tallennus tapahtui reikäkorteille ja nauha-asemille. (Salmela, 2007, 5; DuCharme, 2005.)

Vuosikymmen myöhemmin tietokantojen kehitys loikkasi Apollo-kuuprojektin ja kaupallisen käytön yleistymisen myötä. Kehitettiin ensimmäiset magneettiset levyasemat, jotka mahdollistivat tiedon suoran hakemisen eri tiedostoista. Verkkotietokanta yleistyi ja IBM alkoi kehittämämään hierarkista tietokantaa. Magneettisten levyasemien syrjäyttäessä nauha-asemat kaupallisessa käytössä, alkoi kysyntä hierarkista mallia käyttäville tietokannoille kasvaa. Vuonna 1969 IBM:n IMS (Information Management System) saatiin levitykseen vastaamaan kaupallisia tarpeita. (Salmela, 2007, 7.)

Vuonna 1970 IBM:n tutkija Edgar F. "Ted" Codd julkaisi artikkelin "A Relational Model of Data for Large Shared Data Banks”. Artikkeli oli ensimmäinen teoreettinen perusta relaatiomallista. Ted johti relaatiokantojen testiversioiden kehitystä. Prototyyppiä nimeltä System R. kehitettiin vuodesta 1973 vuoteen 1976. (Salmela, 2007, 9.)

Ensimmäinen IBM:n kaupallinen toteutus DB2 julkaistiin vuonna 1982. Relaatiokantojen suosion johdosta on ilmestynyt useita seuraajia, joista suosituinpina pidetään mm. Access, Oracle, SQL-Server, Ingres, Informix, MySQL (Salmela, 2007, 10). Oraclen yleisyyden vuoksi Suomessa ja Euroopassa, esimerkit ja muu lähempi tarkastelu on pohjattu Oraclen ohjelmistoon ja PL/SQL ohjelmointikieleen.

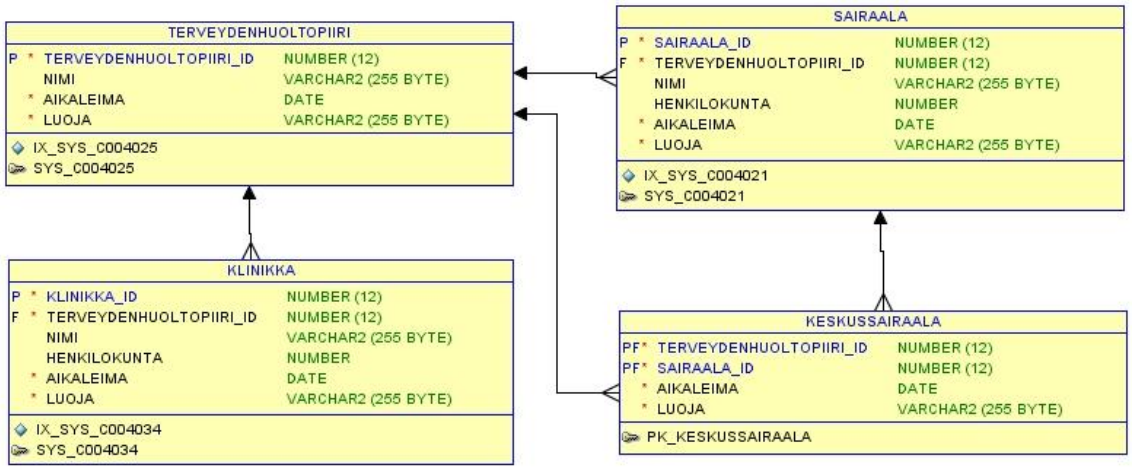
2.2 Rakenne

Relaatiotietokanta koostuu taulukoista, jotka liittyvät toisiinsa loogisesti erilaisia viitteitä hyväksikäyttäen. Taulukot ovat rakenteeltaan kuten taulukkolaskennan taulut. Taulujen rivit on nimetty tietueiksi ja sarakkeet kentiksi. Yksi tietue sisältää yhden tiedon jokaisesta kentästä. Yhdessä taulukossa on aina vähintään yksi avainkenttä, joka määrittelee taulun toimintaa. Avainten mukaan taulun tietueita voi lajitella, yhdistää ja valikoida. (Hovi, 1996, 5-6.)

KLINIKKA_ID	TERVEYDENHUOLTOPIIRI_ID	NIMI	HENKILOKUNTA	AIKALEIMA	LUOJA
6550	40003	Puistolan terveysasema	102	24.05.2010	Opinnäytetyö
6382	40013	Malminkartanon terveysasema	103	24.05.2010	Opinnäytetyö
5824	40017	Kangaslammin terveysasema	104	24.05.2010	Opinnäytetyö
5290	40014	Pihtiputaan terveyskeskus	104	24.05.2010	Opinnäytetyö
4930	40006	Marttilan terveysasema	105	24.05.2010	Opinnäytetyö
5212	40004	Oulunsalon terveyskeskus	105	24.05.2010	Opinnäytetyö
4993	40005	Luumäen terveyskeskus	106	24.05.2010	Opinnäytetyö
5338	40014	Pyhäjoen terveysasema	107	24.05.2010	Opinnäytetyö
4645	40015	Alahärmän terveysasema	107	24.05.2010	Opinnäytetyö
4171	40017	Humppilan terveysasema	108	24.05.2010	Opinnäytetyö
6796	40013	Ullavan tarveysasema	108	24.05.2010	Opinnäytetyö

Kuva1: Klinikkataulun tietue ja kenttä

Hyvä esimerkki avainkentästä on henkilötunnus, koska se on jokaiselle kansalaiselle yksilöllinen. Tauluja voidaan liittää toisiinsa yhteisen kentän avulla. Liitettyjen taulujen avulla voidaan helposti käyttää tietokannan tietoja esimerkiksi yhdistelemällä tai lajittelemalla niitä. Avaimet auttavat pitämään kannan tiedot järjestyksessä ja estävät turhien kopioiden syntymistä. (Holvikivi, 2003.)



Kuva2: Terveydenhuoltopiiri-tietokannan yhteydet

Kuvassa 2 esitetään nuolilla relaatiotietokannan taulujen välisiä yhteyksiä. Pääavaimia merkitsee kirjain P kentän nimen edessä ja viiteavaimia kirjain F. Viiteavain viittaa seuraavan taulun pääavaimeen. Viittaukset koostavat lopulta sovellukselle halutunlaisen tietopaketin, josta nähdään kaikkien taulujen tarjoamat tiedot koskien terveydenhuoltopiiriä.

3 TIETOKANNAN DOKUMENTOINNIN TARVE

Käytössä olevassa tietokannassa on tietokannan rakenteesta huolimatta aina joitakin inhimillisiä virheitä. Käyttäjien syöttämät tiedot vanhentuvat tai syötetyssä tiedossa on yksinkertaisesti kirjoitusvirheitä. Nämä ovat kuitenkin vain tietoja koskevia virheitä, eivätkä suoranaisesti vaikuta tietokannan toimivuuteen hyvin rakennetussa tietokantajärjestelmässä. Tietoa syötettäessä oikein rakennettuun järjestelmään, tietokannan rajoitteet ja hyvin suunnitellut liittymät eivät suvaitse tärkeiden tietojen virheellistä muotoa. Nämä ehkäisevät ongelmien syntymistä ja virheellisten tietojen kerääntymistä tietokantaan. Muutoksia edeltävän tilanteen dokumentointi on tämän vuoksi ensiarvoisen tärkeää.

Tietokannan toiminnot tarvitsevat aika-ajoin päivittämistä uusien tai muuttuneiden tarpeiden vuoksi. Jos päivitykset tehdään ilman ajankohtaista dokumentaatiota, on lähes mahdotonta nähdä yhden muutoksen vaikutusta koko tietokannan toiminnallisuuteen. Lisättäessä uutta toiminnallisuutta, eli uusia tauluja ja kenttiä, jäljelle jää käytöstä poistuneita tai vanhentuneita tietokannan osia. Näitä turhalta vaikuttavia osia kuitenkin tarvitaan kannan tietoliikenneyhteyksien ylläpitämiseen. Vanhat tietokannan osat jätetään usein kantaan suorittamaan toimintojaan. Näin toimitaan, koska uusien yhteyksien ja toimintojen määrittäminen olisi liian työlästä. Huolellisesti muutetut vanhat tietokannan osat eivät ilman dokumentaatiotakaan haittaa toiminnallisuutta tai käyttöä. Ilman dokumentaatiota näitä muutoksia on kuitenkin erittäin työlästä hakea myöhemmin.

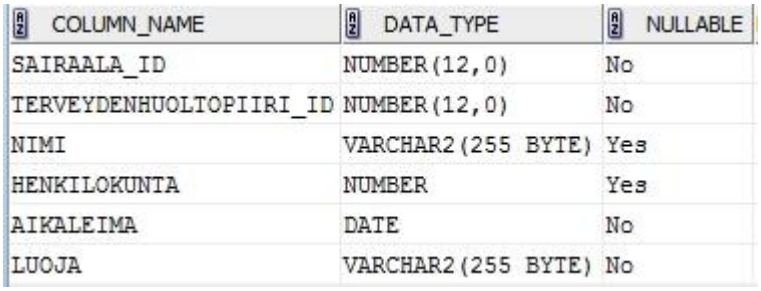
Tietokannan toiminnallisuutta muutettaessa syntyy ristiriitaisuuksia, jotka tarvitsevat selvittämistä ennen muutosten toteuttamista ja käyttöönottoa. Käytöstä poistuneet, turhat ja vanhentuneet tietokannan osat on tunnistettava, jotta saataisiin kuva muutosten todellisesta laajuudesta. Rikkoutunueet ja väärin toteutetut liittymät täytyy kartoittaa, jotta selviää kuinka muutokset vaikuttavat muualle tietokantaan. Kantaan syötettäviä tietoja koskevat selkeästi tarpeelliset rajoitteet ja tietotyypit tulee tarkistaa. Uudet liittymät voivat vaatia eri kentille erilaisia ominaisuuksia kuin alkuperäiset liittymät. Esimerkiksi NOT NULL -rajoitteen lisääminen saattaa olla uuden liittymän kannalta tarpeellinen tietyille kentille tiedon eheyden säilyttämiseksi.

3.1 Kenttien virheelliset tietotyypit ja rajoitteet

Tietokantaa suunniteltaessa aloitetaan yksittäisten taulujen suunnittelusta. Tauluihin lisätään halutut kentät, eli sarakkeet, joille määritellään tietotyypit. Esimerkiksi kellonajoille, päivämäärille ja yleisille numeroille on omat tietotyyppinsä. Tietotyyppi määritellään aina kentän tarpeiden mukaan ja esimerkiksi nimille ja osoitteille riittää tietotyypiksi Teksti.

PL/SQL kielessä kenttiin liitettävät numerotietotyypit ovat: BINARY_INTEGER, DEC, DECIMAL, DOUBLE PRECISION, FLOAT, INT, INTEGER, NATURAL, NUMBER, NUMERIC, PLS_INTEGER, POSITIVE, REAL ja SMALLINT. Merkkijonotietotyypit ovat: CHAR, CHARACTER, LONG, LONG RAW, NCHAR, NVARCHAR2, RAW, ROWID, STRING, VARCHAR ja VARCHAR2. Aikatietotyyppi on DATE. Suurille objekteille on omat tietotyyppinsä BFILE, BLOB, CLOB ja NCLOB. Totuusarvotietotyyppi on BOOLEAN. (Feurstein & Pribyl, 1997, 87-89.)

Tietyille kentille on määritelty NOT NULL -rajoite, jolloin kyseinen kenttä tarvitsee tietotyyppinsä mukaisen tiedon toimiakseen oikein. Juuri näitä rajoitteita käytetään yleisesti tietokannan toimivuuden edistämiseen sekä kentissä, joita tarvitaan tietokannan viite-eheyksien yllläpitämiseen. Pakottamalla kenttien tiedot halutunlaisiksi, ja tarvittaessa pakollisiksi, pystytään vähentämään inhimillisiä syötevirheitä ja estämään tarpeellisten tietojen puuttuminen.



COLUMN_NAME	DATA_TYPE	NULLABLE
SAIRAALA_ID	NUMBER(12,0)	No
TERVEYDENHUOLTOPIIRI_ID	NUMBER(12,0)	No
NIMI	VARCHAR2(255 BYTE)	Yes
HENKILOKUNTA	NUMBER	Yes
AIKALEIMA	DATE	No
LUOJA	VARCHAR2(255 BYTE)	No

Kuva3: Sairaala-taulun kentät

Kuvassa 3 DATA_TYPE määrää kentän tietotyypin ja kentän perässä on tietotyypille mahdollinen tarkenne. Kentässä SAIRAALA_ID esimerkiksi NUMBER (12,0) tarkoittaa, että kentässä saa olla vain lukuja, joiden pituus on maksimissaan 12 numeroa, joista 0 on desimaalipilkun oikealla puolen. Kentän NIMI tietotyyppi VARCHAR2 (255 BYTE) tarkoittaa, että kenttään saa lisätä mitä vain merkkejä, kunhan merkkien kokonaismäärä on alle 255 kappaletta.

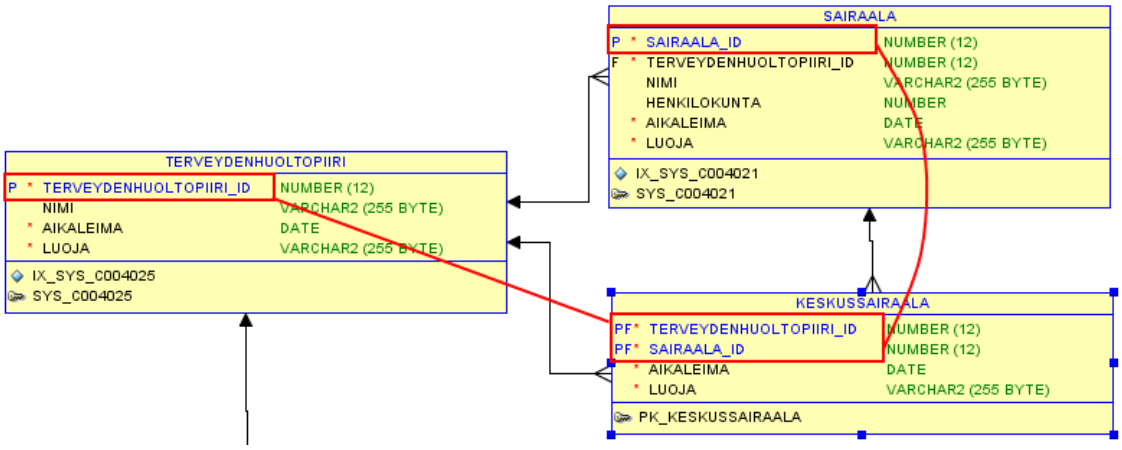
3.2 Käytöstä poistuneet, turhat ja vanhentuneet tietokannan osat

Tietokantaan jää ajan mittaan paljon turhaa materiaalia. Ajankohtainen materiaali haetaan tutkimalla taulun tietueiden aikaleimaa. Aikaleima kertoo milloin tauluun kuuluvaa tietoa on viimeksi muutettu tai milloin kyseinen tieto on lisätty tietokantaan. Tiedon viimeisimmän käyttöajankohdan mukaan voidaan päättää, onko sitä vielä aiheellista säilyttää kannassa.

Tietokannan vaatimukset muuttuvat käyttötarpeiden myötä, jolloin käytöstä usein poistuu vanhoja osia. Tällaiset muutokset tulisi toteuttaa muuttamalla koko kannan toimintaa. Taulujen liitännät tulisi tutkia koko muutosten vaikutusalueelta ja muuttaa uusien käyttötarpeiden mukaisiksi. Usein tällainen muutostyö kuitenkin jätetään tekemättä sen vaatiman suuren työmäärän ja kalliin hinnan vuoksi.

Käytännössä useimmin päädytäänkin vain sulkemaan taulujen tarpeettomat toiminnot ja jättämään taulujen muut toiminnot ennalleen. Yksinkertaistettuna esimerkkinä voidaan ottaa henkilötietotaulu, joka koostuu henkilötunnuksesta, etunimestä, sukunimestä, tuntomerkeistä ja syntymäajasta. Jos tuntomerkkejä ei enää tarvita, poistetaan niiden ylläpitomahdollisuudet ja muutetaan oletusarvoksi tyhjä, eli NULL. Näin taulun muut toiminnot pysyvät samana, vaikka tuntomerkkejä ei enää käytettäisikään.

3.3 Viite-eheys



Kuva4: Keskussairaala-taulun pääavain

Jokaisella taululla on tietorivin yksilöivä uniikki avain. Avain voi koostua yhdestä tai useammasta kentästä. Esimerkin KESKUSSAIRAALA-taulun kohdalla kentät TERVEYDENHUOLTOPIIRI_ID ja SAIRAALA_ID muodostavat yhdessä uniikin pääavaimen. Kun tauluun lisätään tietoa, pääavainkenttään lisätään automaattisesti uusi vapaa avaintieto. Mikäli yksilöivissä avainkentissä on useampaan kertaan samaa tieto, taulu ja avainkenttä on virheellisesti toteutettu. Tästä seuraa, että taulusta haettavaan tietoon ei voida luottaa ja kaikki tietoja käyttävät tietokannan osat ovat epäluotettavia. Tietokantaa muokattaessa tuleekin ensimmäisenä tarkastaa pääavainten yksilöllisyys ja pääavainkentän toiminta.

Jo pienessäkin järjestelmässä käytetään useita relaatioita, eli yhteyksiä. Taulujen ja niiden välisten relaatioiden suunnittelu ja ylläpito ei ole helppoa. Taulujen määrä, yhdistävät kentät ja yhteyksien laatu muodostavat suunniteltavan ja ylläpidettävän kokonaisuuden. Vajavaisesti ja huonosti suunnitellut yhteydet aiheuttavat ongelmia tiedon eheyteen ja pahimmillaan johtavat vakaviin tietokannan tietoja koskeviin virheisiin. Esimerkiksi: "Kahdelle yhdysvaltalaiselle miehelle, joilla oli sama nimi (James Edward Taylor) ja lisäksi sama syntymäaika (23.7.1919), annettiin sama henkilötunnus. Virhe huomattiin 1965, mutta virheen korjaamiseen meni reilut kahdeksan vuotta. Korjaus kesti kauan johtuen mm. siitä, että henkilöiden tiedot olivat useassa eri järjestelmässä." (Paavilainen, 1998, 11).

4 ONGELMIEN ETSIMINEN TIETOKANNASTA

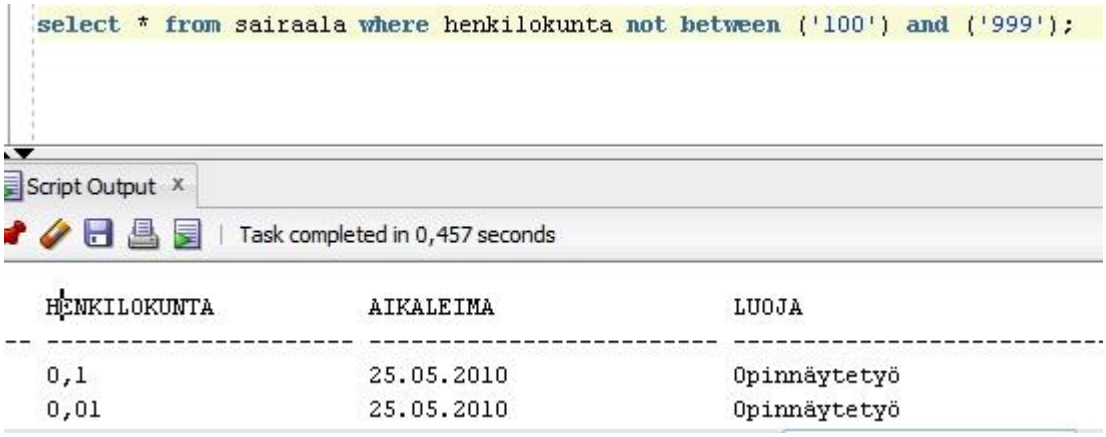
Varsinkin vanhoissa tietokannoissa on oletettavaa, että kannan syötteissä on tapahtunut inhimillisiä virheitä. On myös otettava huomioon vanhan kannan tietoeheyspuutteet, joista myös syntyy virheitä tietokannan tietoihin. Dokumentoinnin haasteita onkin havaita nämä mahdolliset virhetilanteet tietokannan rakennetta ja sisältöä läpikäymällä. Usein tietokannan eri osien käyttötapa tai tarkoitus on saattanut muuttua käytäntöön sopivammaksi tai käyttötapaa on tarkoituksella muutettu. Tietokannan elämisen vuoksi, dokumentointi helpottaa muutosten kartoitusta ja mahdollisten muutosten toteuttamista. Saman elämisen johdosta ongelmien dokumentointi on kuitenkin aina viitteellistä ja viimeinen päätös ongelmien todellisesta olemassaolosta tulee tietokannan ylläpitäjiltä.

Tietokannasta haetaan tietoja erilaisilla hakukyselyillä. Kyselyissä määritellään taulut, kentät ja mahdolliset rajaukset. Rajauksia käytetään suodattamaan tauluista tarvittavat tiedot. Halutessa voidaan esimerkiksi listata vain taulun relaatioiden hakemia tietoja muista tauluista, tai mitä tietoja kyseisestä taulusta muiden taulut käyttävät. Rajaukset ovat välttämätön osa tietokannan ongelmien paikantamisessa kuten myös muussakin tietokannan tietojen hakemisessa. Tietueiden määrä itsessään aiheuttaa usein esteen taulun luotettavalle läpikäymiselle ilman rajauksia.

Peruskysely tapahtuu yksinkertaisella SELECT-lauseella. Tällaisen kyselyn perusrakenne on: SELECT kenttä FROM taulu, ja rajaukset jatkavat kyselyä: WHERE kenttä = 'XX'. SELECT-lauseille voidaan määritellä myös paljon eri toimintoja, jotka yleensä ovat yhteneviä toimintamalliltaan eri kantatyökaluille. Usein nämä toiminnot hieman vaihtelevat kirjoitusasultaan. Oraclen ollessa yleisin SQL-toteutus, on tässäkin tapauksessa esimerkkitilanteissa käytetty Oraclen malleja.

4.1 Kenttien virheelliset tiedot

Vertailemalla taulun kentän sisältämiä tietoja toisiinsa voidaan saada esille selkeästi joukkoon kuulumattomia eriäviä tietoja. Jos tiedetään rajausta helpottavia tekijöitä, niin käytetään niitä hyväksi. Esimerkiksi jos tiedetään, että suomalaisen sairaalan minimimäärä henkilökuntaa on 100 ja ettei Suomessa ole yli 999 henkilön henkilökunnan sairaaloita, voidaan rajata henkilökuntakenttä arvoille 100-999.



Kuva5: Virheelliset henkilökuntamäärät

Listataan HENKILOKUNTA-kentän arvot, jotka eivät kuulu välille 100 - 999 ja löydetään arvot 0,01 ja 0,1. Tällöin on pääteltävissä, että nämä tiedot eivät kuulu joukkoon ja ne kirjataan dokumenttiin. Tekstikenttiä vertaillessa toimitaan hieman eri tavalla.

```
select * from sairaala where nimi not like '%sairaala%';
```

Script Output

Task completed in 0,449 seconds

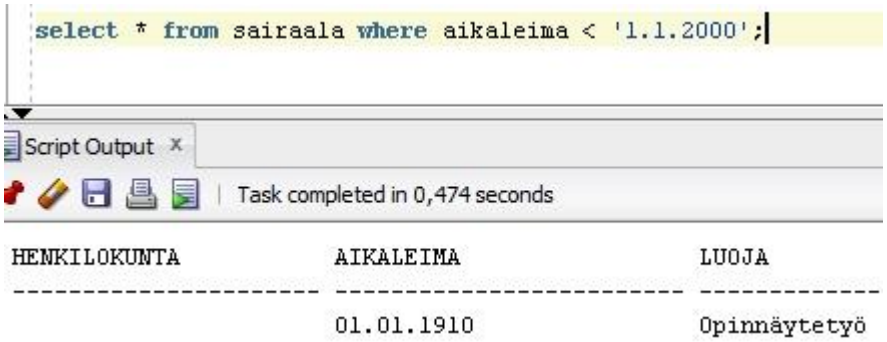
```
*Action:
SAIRAALA_ID          TERVEYDENHUOLTOPIIRI_ID NIMI
-------------------- ----------------------- --------------------------------
4003                 40011                   Helsingin Diakonissalaitos
4027                 40003                   Kruunupuisto
4030                 40019                   Kuopion psykiatrian keskus
4033                 40019                   Kätilöopisto
4042                 40007                   Sairaala Lapponia
4045                 40004                   Lastenklinikka (Helsinki)
4048                 40010                   Lastenklinikka (Oulu)
4051                 40001                   Lastenlinna
4063                 40020                   Nummelan parantola
4069                 40018                   Oulun Diakonissalaitos
4072                 40003                   Paimion parantola
4084                 40016                   Tilkka
6940                 40015                   Hervannan Huoltoasema

13 rows selected
```

Kuva6: Virheelliset nimet

Kyselyllä haettiin kaikki sairaalakentän tietueet, joissa sairaalan nimi ei sisällä merkkijonoa "sairaala". Epäilyttäviä nimiä tarkastellessa voidaan päätellä, että ainakaan "Hervannan Huoltoasema" ei kuulu sairaalatauluun. Muut epäselvät tai epäilyttävät nimet voidaan tässä tapauksessa tarkastaa vaikka Internetistä. Tarkastusten jälkeen kirjataan epäilyttävät tiedot dokumenttiin. Dokumentointi ongelmakohdan osalta on toteutettu. Tämän perusteella tietokannan siivoaminen ja mahdolliset muutokset on kyseisen ongelmakohdan osalta helpompi toteuttaa.

4.2 Vanhentuneet tiedot aikaleimojen perusteella

Tietokannat sisältävät usein tärkeitä tietoja, joita säilytetään kannassa pitkiä aikoja. Käyttötarpeiden muuttuessa jotkin tietueet ja niiden sisältämät tiedot vanhentuvat ja muuttuvat turhiksi. Tarpeettomiksi muuttuneet tiedot saadaan esille tarkastelemalla tietueiden luontiajankohtia.



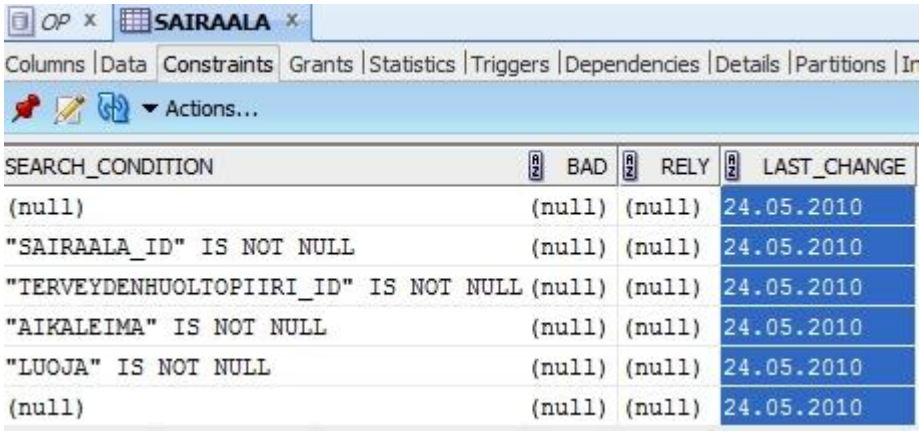
Kuva7: Vanhentunut tietue aikaleimankentän perusteella

Mikäli tietueessa on paljon kenttiä ja tietueita, on syytä tarkistaa, onko jokin kentistä jätetty huomioimatta tarpeettomana. Tällöin tarvitsee tutkia kentät yksitellen.

Kysely: SELECT MAX(*aikaleimakenttä*) FROM *taulu* WHERE *kenttä* IS NOT NULL;

Kyselyllä saadaan viimeisin luontiajankohta, jolloin kyseiseen kenttään on lisätty tietoa. Esimerkiksi, jos tietueessa on tietoja autoista ja siihen on listattu renkaat, polttoaine, iskutilavuus ja korin väri, mutta vuoden 1998 jälkeen ei väriä ole enää jaksettu listata, voidaan olettaa, että kyseinen kenttä on poistunut käytöstä vanhentumalla.

Monissa tietokannankehitysohjelmissa aikaleimat tallennetaan automaattisesti. Vastaavan aikaleiman saa kenttäkohtaisesti selville ohjelman ominaisuuksista. Kuvassa 8 esitetään Oracle SQL Developer -ohjelman kentän viimeisintä muutoshetkeä tarkasteleva ominaisuus.



Kuva8: Tietokantaohjelman aikaleima

Vastaavissa ohjelmissa pidetään usein yllä paljon muutakin tärkeää tietoa, jotta kaikkea ei tarvitse hakea manuaalisesti. Pitää kuitenkin tietää varmasti mistä etsii ja miten. Dokumentaatioon ei huolita ohjelman näytölle syöttäviä virheitä, vaan ainoastaan tietokannan rakenteen ongelmia. Ohjelmia voi toki käyttää apuna, mutta epävarmoissa tilanteissa tarkastukset kannattaa tehdä aina manuaalisesti.

4.3 Tietotyyppien ja rajoitteiden tarpeellisuus

Verrataan kentän tietotyyppejä ja rajoitteita taulun kenttien tietosisältöön ja määritellään täyttävätkö ne tarkoituksensa. Yleisimmät tietokannoissa käytetyt rajoitteet ovat numeroille INT, merkkijonoille VARCHAR ja päivämäärille DATE. Virheelliset tietotyypit saattavat aiheuttaa toiminnallisuudessa näkyviä virheellisiä listauksia, eli tulosteita tietokannasta. Yksinkertaisimmillaan, jos numerokentäksi määritelty kenttä on tietotyypiltään VARCHAR, laskentatoiminnot kyseisestä kentästä eivät toimi.

Rajoitteet kentille ovat joko NULL tai NOT NULL. Rajoitteet ovat tärkeä osa tiedon eheyden ylläpitämistä ja viitteiden toimintaa. Virheellisesti NULL-kentäksi merkitty kenttä, joka toimii viiteavaimena, rikkoo viite-eheyden. Rikkoutunut viite-eheys aiheuttaa virheellisiä listauksia ja pahimmillaan koko toiminnallisuus hajoaa. Virheellisesti NOT NULL -kentäksi määritelty kenttä pakottaa kentälle syötteen, jolloin kentään syötetään aina jotakin ja tietosisältö vaarantuu. Jos NOT NULL -kentän oletussyötettä ei ole määritelty, eikä käyttäjä pääse sitä syöttämään, koko taulun toiminta hajoaa.

4.4 Tarpeettomat kentät käytön perusteella

Tarpeettomat kentät haetaan vastaavalla tavalla kuin virheelliset tiedot, mutta lisäksi tarvitsee ottaa selvää tietojen tarpeellisuudesta. Taulusta tulee selvittää viittaukset ja käydä läpi eri viittauksiin rajatut tiedot.

Kysely: SELECT * FROM sairaala WHERE terveydenhuoltopiiri_id IN (SELECT terveydenhuoltopiiri_id FROM terveydenhuoltopiiri WHERE nimi LIKE '%Hämeen%');

Kyselyllä saadaan taulusta listattua relaation rajaamat tiedot. Eri rajauksilla lista on erilainen ja listat saadaan halutuille rajauksille muuttamalla rajaavasta kentästä haettavia tietoja. Esimerkkikyselyssä määritellään viittaus sairaala-taulun ja terveydenhuoltopiiri-taulun välille. Haettavat tiedot rajataan terveydenhuoltopiirin nimen mukaan. Haetaan ne sairaalat joiden terveydenhuoltopiiri on Hämeessä.

Tyhjät kentät, tai kentät, joissa on vain muutamaa eri arvoa, ovat kenttiä, joista tulee tehdä tarkempi selvitys. Selvityksessä tulee huomioida tietojen rajaukset. Joillain rajauksilla kentät eivät välttämättä ole käytössä, mutta kenttä ei ole tarpeeton mikäli se on käytössä edes yhdellä rajauksella. Jos kenttä on tyhjä kaikilla rajauksilla, tulee selvittää kentän käytön mahdollisuus tietokantaa muokkaavassa sovelluksessa ja listata kenttä mahdollisesti tarpeettomaksi.

4.5 Parametrien etsiminen ja vertailu tietoihin

Verrataan parametroituja tietoja taulun kenttään, johon parametri on määritelty. Parametrit ja niiden arvot määritellään omassa taulussaan, joten parametroidut tiedot tulee hakea parametritaulusta. Tämän johdosta päätaulun parametroitu kenttä hakee viitteillä parametritaulusta tarvittavat tiedot. Parametritietoja käsiteltäessä on huomioitava sekä viite-eheys että tiedon paikkansapitävyys.

Viite-eheyden tarkistaminen tapahtuu samoin kuin muidenkin viittausten tarkistaminen. Ensin tarkastetaan viittaavan ja viitattavan taulun relaatiossa mukana olevat avainkentät. Toiseksi varmistetaan, että kyseisen viittauksen kautta saadaan oikea määrä tietoa. Viitattavasta taulusta tulee myös tutkia mahdolliset jatkoviittaukset muihin tauluihin. Usein muutetuissa tai huonosti suunnitelluissa tietokannoissa nämä relaatioketjut saattavat jäädä osaksi huomioimatta. Viite-eheyden osalta epäilyttävät ja puutteelliset relaatiot tulee dokumentoida.

Kysely: SELECT *parametrin_arvo* FROM *parametritaulu* WHERE *parametrin_arvo* IN (SELECT *kenttä* FROM *taulu*);

Kun ollaan määritelty, että parametroituun kenttään saadaan oikea määrä tietoa, tutkitaan vielä, pitääkö tietosisältö paikkansa. Kentän nimen ja käyttötarkoituksen sekä tietueen muiden kenttien pohjalta saadaan usein hyvä kuva siitä, mitä arvoja kentän tulisi saada. Esimerkiksi, jos terveyslaitoksen tyyppi on parametroitu arvoille "sairaala", "neuvola" ja "terveysasema" ja kentässä onkin jokin muu arvo, viittaus on väärin tai parametroituun kenttään pääsee väärää tietoa. Virheellisten kohtien osalta dokumentoidaan kenttien tiedot.

4.6 Rikkoutuneet ja väärin toteutetut liittymät

Viiteavaimen arvo pitää löytyä viitattavasta taulusta. Haetaan tietueet, joissa viite ja kohde eivät täsmää. Tarkistetaan, että taulu on käytössä ja haetaan viitteet eli viiteavaimet ja niiden kohteet eli kohdetaulun pääavaimet. Mikäli taulussa on useampia viiteavaimia, tarvitsee ne huomioida yhtenä kokonaisuutena. Mikäli molemmat viiteavaimet ovat NOT NULL -rajoitteella, ei voida hyväksyä liittymää, jossa vain toinen viite toteutuu.

Edeltävän tutkinnan perusteella saadaan tarvittavat tiedot jatkotutkimuksille. Mikäli viiteavaimia vastaavia kohdetaulun pääavaimia ei löydy, on liittymä rikki tai väärin toteutettu. Mikäli viiteavain on liitetty kohdetaulun muuhun tietoon kuin pääavaimeen, on viite väärin toteutettu. Jos kohdetaulu on poistettu käytöstä, liittymä on todennäköisesti turha. Jos kohdetaulusta haettava tieto on vanhaa, tutkitaan onko liittymä turha.

Mikäli liittymä on toteutettu useammalla viiteavaimella, tutkitaan voimassaoloehdot ensin. Jos viiteavaimet ovat NOT NULL -rajoitteisia ja yksi tai useampi täyttää jonkin edellä mainitusta kohdista, on liittymä rikki. Jos viiteavaimet eivät ole NOT NULL -rajoitteisia ja yksi tai useampi täyttää jonkin aikaisemmista kohdista, on tarve tutkia haettava tietosisältö. Tietosisällön pohjalta saadaan käsitys liittymän tilasta.

5 LOPUKSI

Tämän opinnäytetyön aiheeseen perehtymisen aloitin hieman reilu vuosi sitten. Työt tietojenkäsittelyalan yrityksessä perehdyttivät minut monimutkaisten tietokantojen rakenteisiin ja vanhojen tietokantarakenteiden ajan kuluessa synnyttämiin ongelmiin. Relaatiotietokantojen rakenteellisten tietojen hakuun vaadittava salapoliisityö pitää työn mielenkiintoisena ja haastavana. Monesti ongelmien löytämiseen liittyvät haasteet osoittautuivat kohtuuttoman paljon aikaavieviksi. Ilman selkeää pohjaa, johon epäilyttäviä tietoja saattoi verrata, jouduin arvaamaan ja dokumentoimaan virheiksi usein aiheellistakin tietoa. Tähän vaikutti suurelta osin selkeän ja yksinkertaisen toimintamallin puuttuminen.

Opinnäytetyön taustalla onkin yksityiskohtaisemmin toimeksiantajalle toteutettu toimintaopas tietojen järjestelmälliseen hakemiseen ja tutkimiseen kohta kohdalta. Toimintamallit listattiin loogiseen järjestykseen. Jotkin tiedoista auttavat toisten tietojen hakemista ja tunnistamista, joten oli ensiarvoisen tärkeää saada kattava tietopaketti aina yhdestä kohdasta ennen toiseen kohtaan siirtymistä. Opinnäytetyössäni toimintamalliesimerkit ovat kuvitteellisesta tietokannasta mutta tekniikat perustuvat siis vahvasti käytäntöön.

Tarkoituksena oli yleistää tietojenhaku- ja tietojenkäsittelytekniikat yleiseen käyttöön ja esimerkeiksi tietokannan dokumentointiin tietokannasta riippumatta. Uskon tämän tavoitteen täyttyneen melko hyvin. Mielestäni suurin hyöty saadaan juuri monimutkaisten tietokantojen dokumentoinnissa. Pyrin kuitenkin pitämään esimerkit mahdollisimman yksinkertaisina, jotta niistä saatava hyöty olisi mahdollisimman kattava. On otettava huomioon, että yksityiskohtaisemmat selvitykset ovat aina tietokantakohtaisia. Uskon kuitenkin, että esittämieni mallien pohjalta on helpompi lähestyä vieraan tietokannan saloja.

LÄHTEET

DuCharme, Bob 2005: "25 years of database history (starting in 1955)". Verkkosivu. http://www.snee.com/bobdc.blog/2005/12/25-years-of-database-history-s-1.html. Viitattu 6.6.2010.

Feurstein, Steven & Pribyl, Bill 1997: Oracle PL/SQL Programming. 2nd Edition. O'REILLY.
Viitattu 6.6.2010.

Holvikivi, Jaana 2003: "Tietojenkäsittelyn perusteet. Tiedonhallinta: relaatiotietokannat". Verkkosivu. http://users.evtek.fi/~jaanah/TkPerusteet/relaatio.htm. Viitattu 6.6.2010.

Hovi, Ari 1996: SQL-opas. Suomen atk-kustannus.
Viitattu 6.6.2010.

Paavilainen, Juhani 1998: SQL-ohjelmointi. Suomen atk-kustannus.
Viitattu 6.6.2010.

Salmela, Eila Helena. 2007: "Tietokantojen historia ennen SQL:ää". Tietojenkäsittelyn historia -seminaarin esitelmä. Helsingin Yliopisto. Verkkosivu. http://www.cs.helsinki.fi/u/kerola/tkhist/k2007/alustukset/tietokannat/. Viitattu 6.6.2010.

LIITE

Liite1: Esimerkkitietokannan rakenteen SQL-developer skripti:

```
--Määritellään hc kierto alkamaan 4000sta ja jatkumaan 3n välein
CREATE SEQUENCE hc INCREMENT BY 3 START WITH 4000;
--Luodaan taulut
CREATE TABLE SAIRAALA(
 SAIRAALA_ID                          NUMBER(12)    NOT NULL PRIMARY KEY,
 TERVEYDENHUOLTOPIIRI_ID      NUMBER(12)    NOT NULL,
 NIMI                                 VARCHAR2(255),
 HENKILOKUNTA                         NUMBER,
 AIKALEIMA                            DATE          NOT NULL,
 LUOJA                                VARCHAR2(255)   NOT NULL);

CREATE TABLE TERVEYDENHUOLTOPIIRI(
 TERVEYDENHUOLTOPIIRI_ID      NUMBER(12)    NOT NULL PRIMARY KEY,
 NIMI                                 VARCHAR2(255),
 AIKALEIMA                            DATE          NOT NULL,
 LUOJA                                VARCHAR2(255) NOT NULL);

CREATE TABLE KESKUSSAIRAALA(
 TERVEYDENHUOLTOPIIRI_ID      NUMBER(12)      NOT NULL,
 SAIRAALA_ID                          NUMBER(12)      NOT NULL,
 AIKALEIMA                            DATE          NOT NULL,
 LUOJA                                VARCHAR2(255) NOT NULL);

CREATE TABLE KLINIKKA(
 KLINIKKA_ID                          NUMBER(12)    NOT NULL PRIMARY KEY,
 TERVEYDENHUOLTOPIIRI_ID      NUMBER(12)    NOT NULL,
 NIMI                                 VARCHAR2(255),
 HENKILOKUNTA                         NUMBER,
 AIKALEIMA                            DATE          NOT NULL,
 LUOJA                                VARCHAR2(255) NOT NULL);

--Määritellään viite- ja pääavaimet
ALTER TABLE
  SAIRAALA
ADD CONSTRAINT
  fk_terveydenhuoltopiiri1 FOREIGN KEY (terveydenhuoltopiiri_ID)
REFERENCES
  terveydenhuoltopiiri(terveydenhuoltopiiri_ID);

ALTER TABLE
  KLINIKKA
ADD CONSTRAINT
  fk_terveydenhuoltopiiri2 FOREIGN KEY (terveydenhuoltopiiri_ID)
REFERENCES
  terveydenhuoltopiiri(terveydenhuoltopiiri_ID);
```

```
ALTER TABLE
  KESKUSSAIRAALA
ADD CONSTRAINT
  fk_keskussairaala1 FOREIGN KEY (terveydenhuoltopiiri_ID)
REFERENCES
  terveydenhuoltopiiri(terveydenhuoltopiiri_ID);

ALTER TABLE
  KESKUSSAIRAALA
ADD CONSTRAINT
  fk_keskussairaala2 FOREIGN KEY (SAIRAALA_ID)
REFERENCES
  SAIRAALA(SAIRAALA_ID);

ALTER TABLE
  KESKUSSAIRAALA
ADD CONSTRAINT
  pk_keskussairaala PRIMARY KEY (terveydenhuoltopiiri_ID, SAIRAALA_ID);

--Määritellään satunnainen avain terveydenhuoltopiirille
CREATE OR REPLACE FUNCTION rand RETURN NUMBER IS
                        random_id TERVEYDENHUOLTOPIIRI.TERVEYDENHUOLTOPIIRI_ID%TYPE;
BEGIN
   SELECT
      TERVEYDENHUOLTOPIIRI_ID INTO random_id
   FROM (SELECT TERVEYDENHUOLTOPIIRI_ID FROM TERVEYDENHUOLTOPIIRI ORDER BY
DBMS_RANDOM.VALUE)
   WHERE ROWNUM = 1;
   RETURN random_id;
END;

--Lisätään satunnainen henkilökuntamäärä 100 ja 1000 väliltä
CREATE OR REPLACE FUNCTION HENKILOKUNTA RETURN NUMBER IS
                        random_id NUMBER;
BEGIN
   SELECT
    CEIL(DBMS_RANDOM.VALUE(100,1000)) INTO random_id
    FROM dual;
   RETURN random_id;
END;
```